ABSTRACT

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

NIND	Natural Image Noise Dataset [3][**nind-ntire**]
BM3D	Block-Matching and 3D filtering [7]
BN	Batch Normalization [12]
cGAN	Conditional Generative Adversarial Network [16]
CNN	Convolutional Neural Network [24]
GAN	Generative Adversarial Network [8]
MSE	Mean Squared Error
RED-Net	very deep Residual Encoder-Decoder Network [15]
ReLU	Rectified linear unit [17]
PReLU	Parametric Rectified Linear Unit [9]
SSIM	Structural Similarity [22]
k	kernel size
p	padding size

1 INTRODUCTION

Photographic image noise occurs as a camera sensor's ISO sensitivity increases to capture an image faster than it would in ideal conditions ("base ISO" sensitivity).[1] A fast shutter speed is often necessary even though there is insufficient light, for instance with handheld photography where a slow shutter speed results in blur caused by the camera shake, or when a dynamic subject results in motion blur. Increasing the ISO setting is akin to linearly amplifying the value measured on each sensor cell. A small initial value that is amplified is less accurate and more prone to errors; this amplified value in turn makes up photographic noise.

Denoising is typically seen as the inverse problem of recovering the latent clean image from its noisy observation [15].

Photographic development usually incorporates a conventional denoising method such as non-local means [4] or wavelets-based methods [**wavelets-denoising**], or a combination thereof, sometimes making use of prior information about a specific sensor's response to different ISO noise values. [29] Block-Matching and 3D filtering [7] (BM3D) is a more powerful denoising method which has been used to benchmark newer techniques [**all-bm3d-benchmarks**], though it is seldom used in off-the-shelf software (presumably because of its high computational cost and the need to specify a γ noise value parameter).

The use of deep learning to solve the denoising problem by directly generating the latent clean image, or in some cases recreating the noise and subtracting it from the observed image [23], has been an active research area. However, while the synthetic results show state-of-the-art performance, testing on real data indicates that neural network-based solutions do not exceed the performance offered by BM3D [18]. It appears that neural networks simply learn the applied noise distribution and that ISO noise may involve additional transformations such as color distortions and loss of detail.

Some specialized work has shown that neural networks obtain state-of-the-art performance when trained with real data [5][25]. We sought to assess the potential of deep learning applied to the denoising problem by expanding on this previous work through a dataset of images produced with various levels of ISO noise. This dataset can be used to train neural network models for general purpose denoising of high quality images.

1 We often make references to ISO noise because increased ISO sensitivity is the main cause of noise, but it should be noted that there are other factors affecting the magnitude of noise acquired by the image sensor.

Our work introduces an open dataset of DSLR-like[2] image sets with various levels of real noise caused by the digital sensor's increased ISO sensitivity. The dataset is large enough to be used for training and varies in content in order to model a great variety of scenes. Each scene was captured at multiple noise levels, with an average of 6 images per set, such that a model may be trained for blind denoising on the base ISO as well as beyond the highest ISO value of the camera by feeding it crops that have a random noise value.

The images in a set are all pixel-aligned. Some of the scenes include multiple ground-truth images which may be sampled at random during training; these would prevent the model from learning to reconstruct the random noise it has seen on one ground-truth, thus making it more difficult to overfit the noise. Overexposed areas are avoided in the ground-truth images because details are lost when the sensor is saturated and this could potentially give an advantage to higher ISO images in which the sensor is not necessarily saturated (notably in ISO-invariant cameras and high ISO pictures that are brightened using software).

The dataset is published in sRGB format on Wikimedia Commons, which is an open-platform that promotes continuous discussion and contribution.

We trained several convolutional neural network architectures presented in the literature, namely DnCNN [23], very deep Residual Encoder-Decoder Network [15] (RED-Net), and U-Net [19], and analysed different proposed methods such as DnCNN [23]'s residual learning.

In most tests a U-Net model was trained with the Natural Image Noise Dataset [3][**nind-ntire**] (NIND) and validated over increasing ISO values; test sets were taken with the same Fujifilm X-T1, as well as a Canon EOS 500D DSLR camera to assess the generalization. We also attempted to combine other datasets with our training data to assess the potential loss in performance caused by generalization. Finally we investigated the use of Generative Adversarial Networks[13] to solve issues such as unnatural face smoothing.

2 We define a DSLR-like camera as one produced with an APS-C (25.1x16.7 mm) or larger sensor such as those present in most DSLR and mirrorless cameras

2 DATASET

The Dataset chapter goes over some existing image noise datasets before delving into the Natural Image Noise Dataset.

2.1 RELATED WORK

The following works feature datasets made of ground-truth / noisy image sets. The static scenes approach is necessary to directly compare the level of degradation using a loss function such as the Structural Similarity [22] (SSIM) index or the Mean Squared Error (MSE).

2.1.1 *Darmstadt Noise Dataset*

The Darmstadt Noise Dataset (DND) [18], containing 50 pairs of noisy-clean images from four cameras, was developed for the purpose of validating denoising algorithms using real data. Synthetic noise is typically used to train and test models, but it had been unclear whether the reported synthetic results translated to real improvements. Plötz and Roth showed that many modern denoising methods do not perform well on real data and that BM3D [7], which was published in 2006, remains one of the best performing methods [18]. RENOIR [2] is a similar dataset that was published prior to the DND; however, Plötz and Roth noted spatial misalignments that reduced its effectivity. We have additionally found that the light sometimes differs between images in the same scene and that some photographs exhibit significant raw overexposure.

2.1.2 *Learning to See in the Dark*

See-in-the-Dark (SID) [5] is an image noise dataset that is large enough for training and, to our knowledge, was used in the first successful attempt at denoising images using real image noise. This dataset focused on very low-light photography where the camera-generated JPEG appears black. The authors used a U-Net network architecture to create an end-to-end RAW-to-JPEG pipeline that produces realistic colors, improving on standard processing and BM3D denoised images which still suffer from color bias at high ISO. Our work differs from SID in that we aimed to train a general purpose (“blind") denoiser rather than one that handles a specific condition, such as extremely low light images. We chose to work in sRGB space because handling the whole RAW-to-sRGB pipeline removes some information which

may otherwise be useful to the author during development. Moreover, one dataset can then be used with different types of color filter arrays.

2.1.3 *Smartphone Image Denoising Dataset*

The Smartphone Image Denoising Dataset (SIDD) [1] is comprised of 10 scenes * 5 cameras * 4 conditions * 150 images, totalling 30000 images. This dataset aims to address the problem of smartphone image denoising, where the small sensor and aperture size causes noticeable noise even in pictures taken at base ISO. Further processing is thus applied to create ground-truth images out of many images. This method of creating ground-truth images is not entirely relevant for denoising images captured with larger sensors because a single image taken at base ISO on a DSLR-like camera is clean enough to work as ground-truth for training purposes.

2.2 NATURAL IMAGE NOISE DATASET

DRAFTy-as-heck sentence

The Natural Image Noise Dataset fills the gap for an image noise dataset of ground-truth - noisy image sets that targets cameras equipped with larger sensors[2] and which is large and varied enough (in both noise values and content types) to train a denoising model for ISO-blind denoising.

FIXME long run-on sentence

Here we outline the physical setup required to capture image sets for the NIND, summarize its content, explain the software processing and validation requirements, and describe its publication aspects such that others that wish to do so may also contribute.

2.2.1 *Capture*

We captured several images per static scene; at least one ground-truth taken with the camera's lowest ISO setting and several images taken with increasing ISO settings and consequent decreasing shutter speed in order to match the original exposure value. Scenes were captured using a camera affixed to a tripod and controlled with a wireless remote control to avoid shifting the setup position. We ensured that the ground was stable, wind would not cause any change in the scene, lighting did not vary between shots, and no area was overexposed in the ground-truth images. Overexposure occurs when the sensor is saturated; on a ground-truth image this would potentially benefit the high ISO images because less light is captured with a faster shutter speed, therefore, the higher ISO images may not be overexposed and thus may contain detail that is not present in the overexposed ground-truth. The aperture remained the same on all shots and the focus was set manually so that it would not automatically adjusted for each frame.

Figure 2.1: Sample ground-truth images from the Natural Image Noise Dataset.

ISO value	100	200	250	320	400	500	640	800	1000	1250	1600	2000	2500	3200	4000	5000	6400	High	Scenes	Images
Fujifilm X-T1		105	11	8	18	13	16	25	16	9	12	9	19	25	17	19	91	123	90	536
Canon C500D	14	10			10			10			9			11				10	11	74
Total	15	116	11	8	29	13	16	36	16	9	22	9	19	37	17	19	91	133	101	616

Table 2.1: Dataset content

A base ISO image (ISO200 on the Fujifilm X-T1) was always taken at least once, along with the camera's highest ISO setting (ISO6400 on the X-T1). Several images were taken with different intermediate ISO values such that the ISO settings varied across each scene. We often also took images that we categorized as "High ISO," which consisted of the highest ISO value and increased shutter speed. "High ISO" images result in dark frames which are then correctly exposed using software. We often tried to match shutter speeds that would be useful to denoise, such as 1/60s for handheld photography, 1/15s for devices equipped with optical image stabilization, and 1/1000s or faster for high-speed photography. We ensured that every ISO value was well represented in order to train models effective at blind denoising. On average, six images were produced per scene and some scenes featured multiple ground truth images which could be used in training to help prevent over-fitting.

2.2.2 *Content*

Many objects were captured in museums where subjects are plentiful (albeit we had to be mindful of copyright restricted material) and for which a denoising application would be highly relevant because indoor handheld pictures require a high ISO sensitivity. However, initial tests found that using images taken only indoors did not provide the variety needed to create a model that generalized well across all conditions, as natural colors were sometimes off in outdoor and brightly lit applications. We thus captured natural objects with vibrant colors (such as food items and plant-life) as well as outdoor scenes where the shutter speed could be taken as fast as 1/13000s using a digital shutter. We made an effort to include some text because it is prevalent, yet we expect a model would not be able to guess how to reconstruct it (Figure **??** shows the resulting denoised text), and we tried to make the images pleasant to look at in order to enhance the

time users would spend looking at them. Most of the NIND images were captured on a Fujifilm X-T1 mirrorless camera, which uses a 23.6 x 15.6 mm X-Trans sensor. A Canon EOS 500D DSLR camera, featuring a 22.3 x 14.9 mm standard Bayer sensor, was used to capture images that could be used to validate the generalization. The content of the dataset is summarized in Table 2.1 and a subset of the X-T1 pictures are shown in Figure 2.1.

2.2.3 *Software processing*

We processed the dataset images using darktable [28] (an open source development software) for raw-to-sRGB development. Our development steps are similar to those we would apply to a standard picture but no sharpening was applied, as this greatly amplifies noise and is typically applied last in the pixel pipeline (we can expect users to apply sharpening to the generated clean image without any perceptible loss). We used darktable's automatic exposure mode to match a fixed percentile target on the histogram and calculate the required exposure compensation for all images in a set. Likewise, we ensured that the white balance was identical and all development steps were copied over to the entire scene. (The exposure percentile and white balance settings are fixed within a scene but vary across different scenes.) The raw overexposed indicator was used to verify that no overexposed areas were present in the base ISO image and if any were detected then it was cropped out or the scene was discarded. The images were visually inspected to detect slight variations in light, the introduction of foreign objects such as insects, or any movement, which also resulted in cropping or discarding images. The ground-truth images must be at least as sharp as their noisy counterparts; this is sometimes not the case due to slight movements on longer exposures. The remaining images were saved in either high-quality (98 to 100) 8-bit JPEG or lossless 16-bit PNG.

The last step of development is to use Hugin's align_image_stack tool [11] to ensure that all images in a set are perfectly pixel-aligned. The tool will usually return the same image size as the input, in which case the whole image set can safely be used. When a difference is detected then the tool will automatically align the set and we visually analyzed whether the result was acceptable or the movement caused a change in perspective, in which case the outlier images were discarded. Some noisy images cannot easily be matched to the scene; possible solutions are to denoise these images in order to check the alignment or to take a cleaner image afterward and assume that the middle images are consistent with the previous and next ones.

2.2.4 *Publishing*

The Natural Image Noise Dataset is published on Wikimedia Commons (https://commons.wikimedia.org/wiki/Natural_Image_Noise_Dataset), an online repository of free-use images and other digital media. Wikimedia Commons hosts media content for all Wikimedia projects and its scope is limited only by the content having some educational value (broadly meaning "providing knowledge; instructional or informative"). As such, we believe it is a fitting platform for the publication of a research dataset.

One key advantage of using Wikimedia Commons is its collaborative aspect. Anyone is allowed to add images to the dataset, modify existing images (for example to fix a spatial misalignment), and discuss the content (through the discussion page provided for each file, category, and the dataset itself). The collaborative aspect also includes a "Quality images candidates" page [6] where users assess the technical quality of a submitted image and may promote it to a "Quality image" standing. Many of the ground-truth images have gone through this process and were promoted through human assessment. The same process was also used to validate the trained model, which ended up in a positive assessment since even the denoised dynamic ISO6400 picture presented in Figure **??** was among the promoted images.

On the technical side, Wikimedia Commons preserves images as they are uploaded; JPEG images are not recompressed, 16-bit lossless TIFF and PNG images are allowed, and the metadata is kept. Thumbnails are generated and images may be visualized before being downloaded in full resolution, and the download may include a select subset instead of the whole dataset. A customizable download script is provided on the dataset's page for convenient retrieval. Even though files can be overwritten, every file uploaded on Wikimedia Commons is kept forever therefore specific snapshots of the dataset can be made by including the files' revision in the download script and getting a specific version ID (or commit hash) of the download script.

It is also important to note the usefulness of the data outside of a denoising dataset context. Many images, such as those depicting artifacts displayed in churches and museums, have encyclopedic value and the ground-truth images present in our dataset are of higher quality than most previously available images depicting such artifacts. By publishing the dataset on Wikimedia Commons, these ground-truth images may be used in Wikipedia articles directly[1]. This may be a motivating factor to those wishing to contribute.

1 E.g. Bobo people, Bombardment of Brussels, Dengese people

3 NEURAL NETWORK ARCHITECTURES

This chapter introduces the notion of convolutional neural networks and their components, then goes over the specific network architectures we have worked with.

3.1 CONVOLUTIONAL NEURAL NETWORKS

We use deep convolutional neural networks which are neural networks based on the multilayer perceptron; that is, input data from one layer is sent to the neurons of the next layer and a dot product is performed between that input data and the next layer's learned weights. This process goes on for each layer. The main difference between a multilayer perceptrons and a Convolutional Neural Network [24] (CNN) is that a CNN is not fully connected (that is neurons in one layer only receive a subset of the previous layer's neurons), instead, a CNN takes advantage of the local spatial coherence of the input images to share weights and thus reduce the number of shared parameters.

The first and last layer have the same shape as the input image, that is its height, width, and 3-channels representing the RGB color-space. Most layers are hidden layers between the first and last one and contain many more channels (also called feature maps). Hidden layers are usually convolution (or transposed convolution) layers, activation layers, or pooling layers.

A loss function is applied to the output of the last layer, its gradient with respect to the network's weights is calculated with the back-propagation algorithm, and the Adam gradient-descent algorithm[14] adjusts the weights at the end of each batch training iteration. The learning rate needs to be fine-tuned to optimize fast learning without over-fitting.

3.1.1 *Layers*

Convolution layers apply C convolutions (also called filters) to every kernel size $(k) * k$ area in the previous layer's width and height[1], taking in every channel and returning one value per convolution. Convolutions have a receptive field that is determined by k (typical values for k are 3 and 5). The next layer's width and height is thus reduced unless padding is applied. Given an input layer's width or height dimension d and padding size (p), a convolution layer's output size can be calculated as $d - k + 2p + 1$. Each filter is weighted and

1 assuming a stride of 1

weights are learned parameters. The same stack of filters is applied on every $k * k$ area of a given layer, therefore CNN have largely reduced number of weight parameters.

Activation layers introduce non-linearity, they apply an activation function to every value in the previous layer and the resulting layer has the same shape as its input layer. We use the Sigmoid ($\phi(z) = \frac{1}{1+e^{-z}}$) and Rectified linear unit [17] (ReLU) ($R(z) = \max(0, z)$) activation functions.

3.1.2 *Loss function*

The loss function assigns a score the the generated images with respect to the provided ground-truth. Loss functions need not to simply compare the difference between generated and ground-truth pixels because there is no one-to-one mapping between clean and noisy images, therefore a loss function which only focuses on the difference between pixels favors denoised images whose pixels average the many acceptable values; i.e. the network is trained to generate blurry images. [13]

The MSE (or $\ell 2$) measures the average squared difference between the generated and ground-truth pixels, it is a simple yet effective loss function. Similarly the $\ell 1$ loss measures the sum of the absolute differences between ground-truth and generated pixels, this typically favors blurrier results. The SSIM index is another metric which puts weights on structural information that is more likely to be perceived in images, namely local changes in luminance, contrast, and structure. A loss may also be learned such as the discriminator in a Generative Adversarial Network [8] (GAN).

Mixing different losses often appears to yield better results [26], although it is sometimes challenging to combine them because they are often scaled differently (and training may suffer a performance penalty when having to compute multiple losses). Another valid approach is to switch loss function once convergence has been attained with one. [26] We typically use the SSIM index in our standard models and a combination of loss functions in the more challenging GAN.

TODO: batch normalization

3.2 NETWORK ARCHITECTURES

3.2.1 *DnCNN*

DnCNN is a simple flat architecture CNN architecture introduced in [23] to denoise images with residual learning. The network performs end-to-end denoising using only a convolution layer followed by Batch Normalization [12] (BN) and a ReLU activation layer at each depth level. The dimensions of the layers remain constant because the convolutions are padded appropriately ($p = \frac{k-1}{2}$).

The main contribution introduced in [23] appears to be the use of residual learning to speed up learning by modeling and subtracting the residual noise rather than generating a clean image. The use of a simple DnCNN architecture may have been introduced to emphasize the effectiveness of residual learning. DnCNN is the first network we experimented with, and although most of our experiment focus on the subsequent networks, we tested the author's residual learning theory with our dataset of ISO noise and all three networks.

3.2.2 *RED-Net*

The RED-Net architecture is a residual encoder-decoder network. The first half of the network encodes the image using convolutions without padding such that the layers' dimensions continuously decrease, then the second half of the network (decoder) is made of transposed convolution layers that increase the layers' dimensions up to the original image's height and width. A ReLU activation layer is placed between every (de)convolution layer. What is referred to as a residual network differs from the residual learning of [23], a residual network has skip connections which in the case of [15] are placed every two convolution layers and connect them to transposed convolution layers of matching dimensions. These skip connections add details that may have been lost in the encoding process to the output of the transposed convolutions.

3.2.3 *U-Net*

The U-Net architecture is an encoder-decoder with skip connections originally designed for image segmentation. Its use in an image generator is possible by using the number of input color channels as output features and upsampling the resulting image. Each U-Net block applies two 3x3 convolutions followed by a parameter-free pooling operation that downsamples the image by a factor of two. The first block has 64 filters and the number of filters doubles after every pooling operation until there are 1024 filters. From this point the downsampling pooling operations are replaced with 2x2 upsampling transposed convolutions and the number of filters get halved every time, but the convolutions remain (as opposed to using transposed convolutions) so the layers' size remain smaller on the upsampling stage than it is originally. (This design results in corrupted borders which need to be discarded.) The U-Net architecture uses significantly less resources than conventional encoder-decoder networks and it is able to capture details at multiple frequencies.

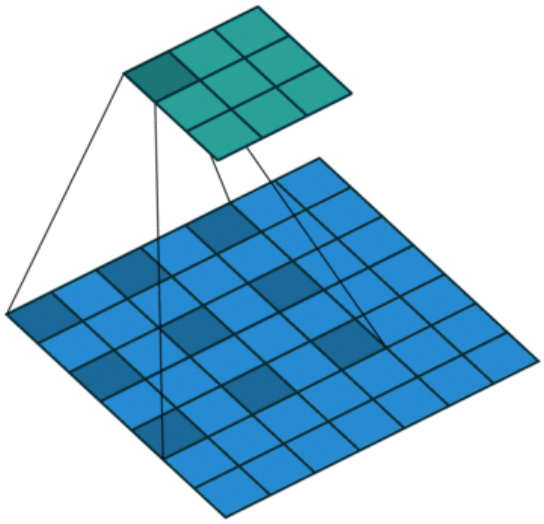

Figure 3.1: Dilated convolution (dilation=2)

3.2.4 *HulbNet*

We designed a multi-scale architecture somewhat similar to U-Net whose aim is to be more adapted to the denoising problem at hand. HulbNet uses the following concepts:

- **Transposed convolution**: The blocks present in the upsampling stage use transposed convolutions rather than standard convolution in order to restore the original layers' size. Transposed convolutions are used in RED-Net as well.
- **Dilated convolution** adds space between each element of the convolution filter as shown on Figure 3.1. This allows the receptive field to grow and the output layer's dimensions to shrink without increasing the number of parameters and computation steps.
- **Stride** is the xy space the convolution filter moves between each step in a layer. The default convolution has a stride of one such that each element is seen three times in a 3x3 convolution. We apply convolution with a stride of three instead of using a 2x2 pooling operation to downsample the image between blocks. The resulting feature maps are downsampled by a factor of three with learnable parameters, as shown on Figure 3.2.
- **Dense connections**: HulbNet uses concatenated filters similar to the concept introduced in DenseNet [10]. Residual connections such as those used in [15][19] are element-wise sum of the feature maps, as such the feature maps are combined and the network

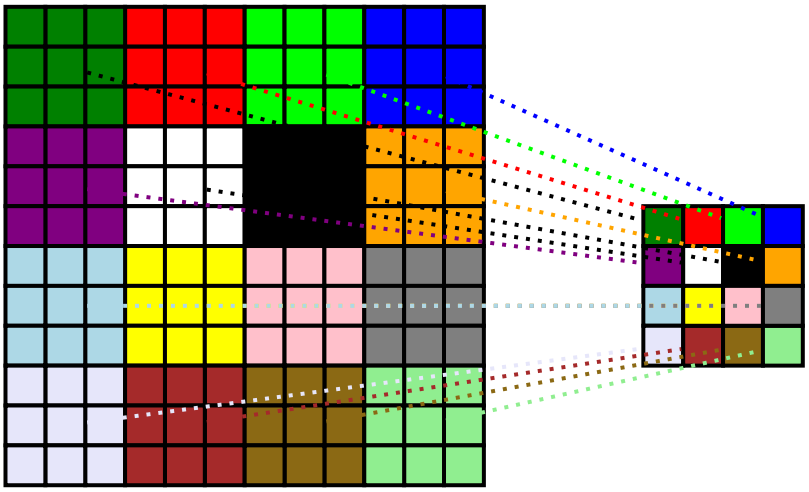

Figure 3.2: 3x3 convolution with stride=3

can no longer chose individual elements to learn from. Dense connections on the other hand form concatenated features, that is the receiving filter can learn from any of the features separately and features can bring new information that does not necessarily need to be strongly correlated or have matching depths; features may therefore come from filters with different receptive fields and in different parts of the network.

HulbNet has two main types of convolution blocks; standard convolutions are made up of two series of standard 3x3 convolutions while dilated convolution blocks have one convolution with dilation set to two. These blocks have the same effective receptive field with the height and width of the feature maps being reduced by two after each block, and so they are both applied then concatenated as the input to the next layer. The motivation behind this combination is to capture features from multiple resolutions as the dilated convolution is expected to focus more on low-frequencies and repeated standard convolutions would capture the finer details. The first layer has an additional dilated convolution with dilation of five to capture an especially wide window of 10×10.

After two blocks of each type, the feature map is downscaled by a factor of three with the aforementioned strided convolution. The strided convolution is expected to result in the most appropriate type of pooling (or combination thereof), and using a stride of 3 (which reduces the dimensions by the same factor) results in each pixel being processed exactly once therefore avoiding checkrboard patterns which can occur when some lines are processed more than others.

The blocks and downscale operations are repeated three times (with minor adjustments based on the intended input size) resulting in a single feature stack. The network is then mirrored using transposed convolutions that replace the convolutions to grow back to the original

image size. All concatenated features in the downscaling part of the network are concatenated into the input of each upscaling block. The last block features a bottleneck 1x1 convolution which reduces the number of features down to three channels. Each convolution is followed by a Parametric Rectified Linear Unit [9] (PReLU) activation. Batch Normalization [12] was not applied because experimental results showed worse performance using it in denoised image generation.

HulDisc is a matching discriminator network which follows the same structure as the aforementioned HulbNet but end in the middle with a single probabilistic feature.

4 GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks are pairs of competing networks; a generator and a discriminator contest each other in a zero-sum game. The goal of the generator is to fool the discriminator by generating content which cannot reliably be differentiated from real ground-truth content. The discriminator attempts to tell generated content apart from real data. Each network is trained as the other's loss function (or one thereof).

The main advantage of a Generative Adversarial Network [8] is that the loss function is learned; it does not need to be defined and it can exceed the performance obtained with a conventional loss that is a limiting factor. The generator can learn to restore high-frequency details which cannot be determined from the noisy observation alone (and that are not necessarily visible in the ground-truth) instead of averaging the many possible solutions.

Moreover, data may be added without a matching pair in some scenarios. This does not work with Conditional Generative Adversarial Network [16] nor can it be combined with other loss functions (combining losses is often necessary to ensure that the result matches and that the details are not overdone), but it can be of great usefulness to further train an existing network to perform better on specialized tasks.

4.1 TYPES OF GANS

By input:

- GAN A GAN is the simplest approach introduced in [8]. A generator is called with a given input, then the discriminator trains on a mini-batch made of the generated data and another mini-batch containing ground-truth data.

  A GAN does not necessarily need clean-noisy pairs of images because it only receives one image at a time. However, this feature may lead the generator to learn a mapping which has the features required to trick the discriminator without being related to the input image. For this reason we often combine the discriminator loss with a conventional loss function such as SSIM or L1 (and these losses do require a pair of clean-noisy images). The GAN could still be used for semi-supervised learning by training with conventional losses and a paired dataset until satisfying performance is met, then training further with a non-paired dataset using the GAN only. This could be beneficial for

specialized images which would be tricky to include in a paired dataset, such as those depicting moving subjects or human faces.

While it may seem counter-intuitive to train the discriminator with mini-batches of only one label at a time, this phenomenon may be explained with the use of Batch Normalization which keeps running statistics and normalizes the activations on the given batch, it could therefore perform better given low-variance input data. [20][12]

- Conditional Generative Adversarial Network [16] (cGAN) Conditional **GANs!**s are GANs which take two inputs, the generated data and typically a label that describes it. An approach often used in image generation is to concatenate the corrupted image with the generated input and feed these image pairs as the input data rather than using a label. [13][27][21]. The benefit of potentially semi-supervised learning is no longer present but the discriminator is built-in with a correlative loss and the network is less likely to require a conventional loss function.

CycleGAN[27] is a popular approach which enforces cyclic consistency, that is, the network must be able to regenerate the corrupted observation back from the generated one. We believe this approach would not function well for the problem of image denoising because of the random nature of noise, whereas CycleGAN applications tend to have simple outlines as the corrupted observation.

By loss:

- **LSGAN!** (**LSGAN!**)
- **DCGAN!** (**DCGAN!**)

4.2 CHOSEN GAN

cGAN and GAN PatchGAN PatchGAN already work on small 128x128 crops (of which the discriminator typically only sees 112x112 because we do not discriminate against bad borders) The generator is the same UNet architecture used in previous

TODO Move this to later discussion

LSGAN

4.3 METHOD

The discriminator is trained on the generated input as soon as the network is initialized, but the generator uses only conventional loss functions (namely SSIM and L1) until a set SSIM threshold is met. From this point the loss function is a weighted combination of the discriminator's MSE loss (0.5), the SSIM index (0.45) ensures stability

and image quality, and the L1 loss (0.05) provides some smoothness to overcome some of the excessive high-frequency details caused by the discriminator.

Balanced initialization is crucial as experimental work has shown that using the discriminator too soon results in a generator whose only focus is to trick the discriminator and the image quality suffers to the point where the generator loses and its gradient vanishes. Starting off with a well-trained generator on the other hand gives no chance to the discriminator whose best bet is to be forever indecisive.

The network is still highly susceptible to failure after initialization. Common issues are overconfidence (with eventual mode collapse where the network outputs the same result), vanishing gradients, and indecision, therefore a good balance must be maintained all along the training process. We provide the discriminator's MSE loss with noisy labels (+/- 0.03 probabilities) to prevent overconfidence. There is typically a 0.33 training ratio between the discriminator and the generator, the imbalance ensures that the two networks are not constantly trying to overcome each other's latest update. This ratio is lowered to 0.1 when the discriminator gets too accurate (95%), and it is increased to 0.9 when the discriminator has approximately 50% or below accuracy. (50% is the ideal target but it often indicates an indecisive discriminator.)

Algorithm 1 (c)GAN training procedure

```
for all clean_batch, noisy_batch ∈ Dataset do
  ADAM_optimizer_D.clear_gradient()
  ADAM_optimizer_G.clear_gradient()
  {Generate a denoised ("fake") images batch}
  generated_batch ← network_G(noisy_batch)
  {Train the discriminator}
  if network_D.is_conditional then
    fake_minibatch ← concatenate(
      remove_borders(noisy_batch),
      remove_borders(generated_batch).detach_from_graph())
    real_minibatch ← concatenate(
      remove_borders(noisy_batch),
      remove_borders(clean_batch))
  else
    fake_minibatch ← remove_borders(generated_batch).detach_from_graph()

    real_minibatch ← remove_borders(clean_batch)
  end if
  predictions_on_real_data ← network_D(real_minibatch)
  loss_real_D ← MSE(
    predictions_on_real_data,
    generate_noisy_probabilities(true))
  loss_real_D.backpropagate()
  predictions_on_fake_data ← network_D(fake_minibatch)
  loss_fake_D ← MSE(
    predictions_on_fake_data,
    generate_noisy_probabilities(false))
  loss_fake_D.backpropagate()
  if discriminator_learns then
    {Whether the discriminator learns depends on randomness and
    a ratio determined by its previous performance}
    ADAM_optimizer_D.update_weights()
  end if
  {Train the generator}
  loss_G_GAN ← MSE(
    predictions_on_fake_data,
    generate_noisy_probabilities(true))
  loss_G_SSIM ← SSIM(
    remove_borders(generated_batch),
    remove_borders(clean_batch))
  loss_G_ℓ1 ← ℓ1(
    remove_borders(generated_batch),
    remove_borders(clean_batch))
  loss_G_weighted ←
    loss_G_GAN ×(1− weight_loss_SSIM − weight_loss_ℓ1)
    + loss_G_SSIM × weight_loss_SSIM
    + loss_G_ℓ1× weight_loss_ℓ1
  loss_G_weighted.backpropagate()
  ADAM_optimizer_G.update_weights()
end for
```

5 RESULTS

6 CONCLUSION

6.1 FUTURE WORK

- Discriminator loss needs to be much better - multi-scale (pix2pixHD)
  - train a general network with specialized dataset

BIBLIOGRAPHY

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. "A High-Quality Denoising Dataset for Smartphone Cameras." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[2] Josue Anaya and Adrian Barbu. "RENOIR – A dataset for real low-light image noise reduction." In: *Journal of Visual Communication and Image Representation* 51 (Sept. 2014). DOI: 10.1016/j.jvcir.2018.01.012.

[3] Benoit Brummer. *Natural Image Noise Dataset - Wikimedia Commons*. https://commons.wikimedia.org/wiki/Natural_Image_Noise_Dataset.

[4] A. Buades, B. Coll, and J. . Morel. "A non-local algorithm for image denoising." In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. 2005, 60–65 vol. 2. DOI: 10.1109/CVPR.2005.38.

[5] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. "Learning to See in the Dark." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[6] *Commons:Quality images candidates - Wikimedia Commons*. https://commons.wikimedia.org/wiki/Commons:Quality_images_candidates.

[7] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. "Image denoising with block-matching and 3D filtering." In: *Proceedings of SPIE - The International Society for Optical Engineering* 6064 (Feb. 2006), pp. 354–365. DOI: 10.1117/12.643267.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[9] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[10] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. "Densely Connected Convolutional Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[11] *Hugin - Panorama photo stitcher*. http://hugin.sourceforge.net/.

[12] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632. URL: https://doi.org/10.1109/CVPR.2017.632.

[14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6980.

[15] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections." In: *Proc. Advances in Neural Inf. Process. Syst.* 2016.

[16] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets." In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[17] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 978-1-60558-907-7. URL: http://dl.acm.org/citation.cfm?id=3104322.3104425.

[18] Tobias Plotz and Stefan Roth. "Benchmarking Denoising Algorithms With Real Photographs." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[19] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved Techniques for Training GANs." In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2234–2242. ISBN: 978-1-5108-3881-9. URL: http://dl.acm.org/citation.cfm?id=3157096.3157346.

[21] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs." In: *CoRR* abs/1711.11585 (2017). arXiv: 1711.11585. URL: http://arxiv.org/abs/1711.11585.

[22] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity." In: *IEEE Transactions on Image Processing* 13 (Apr. 2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[23] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising." In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[24] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. "Parallel distributed processing model with local space-invariant interconnections and its optical architecture." In: *Appl. Opt.* 29.32 (1990), pp. 4790–4797. DOI: 10.1364/AO.29.004790. URL: http://ao.osa.org/abstract.cfm?URI=ao-29-32-4790.

[25] Yide Zhang, Yinhao Zhu, Evan Nichols, Qingfei Wang, Siyuan Zhang, Cody Smith, and Scott Howard. "A Poisson-Gaussian Denoising Dataset with Real Fluorescence Microscopy Images." In: *CoRR* abs/1812.10366 (2018). arXiv: 1812.10366. URL: http://arxiv.org/abs/1812.10366.

[26] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. "Loss Functions for Image Restoration With Neural Networks." In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57. ISSN: 2333-9403. DOI: 10.1109/TCI.2016.2644865.

[27] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss." In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[28] *darktable*. https://www.darktable.org/.

[29] *profiling sensor and photon noise | darktable*. https://www.darktable.org/2012/12/profiling-sensor-and-photon-noise/. Dec. 2012.